(2)

IDA PAPER P-1925

PROCEEDINGS OF THE CAIS/CIG/SEI WORKSHOP

Clyde Roby

January 1986

Prepared for
Ada Joint Program Office

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772

89

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, or (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Papers

Papers normally address relatively restricted technical or policy issues. They communicate the results of special analyses, interim reports or phases of a task, ad hoc or quick reaction work. Papers are reviewed to ensure that they meet standards similar to those expected of refereed papers in professional journals.

Documents

IDA Documents are used for the convenience of the sponsors or the analysts to record substantive work done in quick reaction studies and major interactive technical support activities; to make available preliminary and tentative results of analyses or of working group and panel activities; to forward information that is essentially unanalyzed and unevaluated; or to make a record of conferences, meetings, or briefings, or of data developed in the course of an investigation. Review of Documents is suited to their content and intended use.

The results of IDA work are also conveyed by briefings and informal memoranda to sponsors and others designated by the sponsors, when appropriate.

The work reported in this document was conducted under contract MDA 903 84 C 0031 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that agency.

This paper has been reviewed by IDA to assure that it meets high standards of thoroughness, objectivity, and sound analytical methodology and that the conclusions stem from the methodology.

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	**1b RESTRICTIVE MARKINGS**
2a SECURITY CLASSIFICATION AUTHORITY	**3 DISTRIBUTION/AVAILABILITY OF REPORT** Public release, unlimited distribution: 23 December 1988
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) IDA Paper P-1925	**5 MONITORING ORGANIZATION REPORT NUMBER(S)**
6a NAME OF PERFORMING ORGANIZATION Institute for Defense Analyses **6b OFFICE SYMBOL** IDA	**7a NAME OF MONITORING ORGANIZATION** OUSDA, DIMO
6c ADDRESS (City, State, and Zip Code) 1801 N. Beauregard St. Alexandria, VA 22311	**7b ADDRESS (City, State, and Zip Code)** 1801 N. Beauregard St. Alexandria, VA 22311
8a NAME OF FUNDING/SPONSORING ORGANIZATION Ada Joint Program Office **8b OFFICE SYMBOL (if applicable)** AJPO	**9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER** MDA 903 84 C 0031
8c ADDRESS (City, State, and Zip Code) 1211 Fern Street, Room C-107 Arlington, VA 22202	**10 SOURCE OF FUNDING NUMBERS** PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. T-D5-305 / WORK UNIT ACCESSION NO.
11 TITLE (Include Security Classification) Proceedings of the CAIS/CIG/SEI Workshop (U)	
12 PERSONAL AUTHOR(S) Clyde Roby	
13a TYPE OF REPORT Final **13b TIME COVERED** FROM _____ TO _____	**14 DATE OF REPORT (Year, Month, Day)** 1986 January **15 PAGE COUNT** 95
16 SUPPLEMENTARY NOTATION	

17 COSATI CODES			**18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)**
FIELD	GROUP	SUB-GROUP	Ada; CAIS; APSE; Programming Support Environments; CAISWG; CAIS security requirements; Input/Output packages; CAIS and the Space Station; MIL-STD-CAIS.

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of this IDA Paper is to document the general areas of discussion on CAIS Security requirements, the CAIS and other environment development efforts (including PCTE, the WIS Operating System, UNIX, etc.), Input/Output packages in the CAIS, the CAIS package LIST_UTILITIES, Distributed CAIS, CAIS and the Space Station, and a list of issues brought forth by the implementors.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	**21 ABSTRACT SECURITY CLASSIFICATION** Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Mr. Clyde Roby	**22b TELEPHONE (Include area code)** (703) 824-5536 **22c OFFICE SYMBOL** IDA/CSED

IDA PAPER P-1925

PROCEEDINGS OF THE CAIS/CIG/SEI WORKSHOP

Clyde Roby

January 1986

INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 84 C 0031
Task T-5-305

Table of Contents

1.0 Introduction

1.1 Background

1.1.1 History of the CAIS

The Ada[1] language was adopted as a standard by the American National Standards Institute in early 1983. [1]

In parallel with the latter stages of the design of the Ada language, an effort was begun to identify "the relationships of the parts of an integrated Ada Program Support Environment (APSE)." [11] This effort resulted in the "Stoneman" document which is formulated as a requirements specification for an APSE.

"In December 1980 the Under Secretary of Defense for Research and Engineering [OUSRDE] established the Ada Joint program Office (AJPO) to manage DoD efforts for the introduction, implementation, and life cycle support of Ada. . . . In order to coordinate APSE developments, the AJPO obtained a Memorandum of Agreement (MOA) with the three services at the Assistant Secretary level. The tri-service agreement focused on the need to develop a means by which tools and databases can be readily transported across service-specific APSE implementations. The concept of the KAPSE, as articulated in the APSE STONEMAN document, is the focal point for tri-service commonality." [7]

In January 1982, an evaluation team called the KAPSE Interface Team (KIT) met for the first time with the long term objective of defining "requirements for interoperability and transportability among KAPSE's, followed by guidelines and conventions for achieving them. . . . [These] will evolve into standards which, when followed, will ensure the ability of APSEs to share tools and data bases. . . . [The KIT was supplemented] with a team of representatives from industry and academia [called the KAPSE Interface team from Industry and Academia (KITIA)] . . . to provide the KIT with a broad base of inputs, reviews, and advice from the technically qualified talent in industry and academia." [7] Prior to the creation of the KIT/KITIA there were two service-specific APSE implementations under construction: the Ada Language System (ALS) for the Army and the Ada Integrated

---

[1]Ada is a registered trademark for the U. S. Government, Ada Joint Program Office.

Environment (AIE) for the Air Force. The capabilities provided by the ALS and the AIE were derived from STONEMAN, but the KAPSE interfaces (primitive operations and abstract data types not defined in the Ada language, but necessary for the execution of APSE tools) differed significantly. Thus, software tools using the KAPSE interfaces of one APSE could not be transported to the other APSE without changing the calls to the KAPSE interfaces. Additionally, the differences between the capabilities provided by the KAPSE interfaces of the two APSEs potentially made transporting a software tool difficult.

A good history of the background on this effort was generated by Tim Harrison of Texas Instruments and appears in [4].

The proposed "Military Standard Common APSE Interface Set (CAIS)" [2] was delivered to the AJPO on 31 January 1985. The proposed MIL-STD-CAIS is currently being reviewed by the services for adoption as a DoD standard.

1.1.2 Purpose of the CAIS

The successful Ada standardization effort has resulted in a language definition which will serve to increase the host and target transportability of software written in the Ada language. Many computer programs, however, have complex interactions with their environment including the host operating system, run-time libraries, other programs, external data, terminals, and users. The Ada language definition does not extend beyond the execution of an individual program.

To support the transportability of Ada programs across environments, the CAIS was developed to define a standard for interfaces between Ada programs and their programming support environments. The CAIS defines a set of interfaces for Ada programs which, if uniformly employed, will enhance the transportability of tools for software development and management. These interfaces are specified in the Ada language with accompanying English description of semantics.

1.1.3 CAIS Implementation efforts

In the past, there have been several CAIS prototyping efforts:

a. TRW was funded by the government to do a partial implementation for the study of the CAIS interfaces; this work included some tool rehosting. The host for this effort was UNIX/ARCTURUS on a VAX.

b. Gould has been doing a full implementation (including tools and rehosts) on their MPX and UTX operating systems on an SEL system. This internal effort will be described more thoroughly in these proceedings.

c. MITRE (funded internally) is doing a partial implementation (including tools) on ULTRIX on the VAX. More about this effort is described in these proceedings, too.

d. Intermetrics (funded by the government) had been working on a partial implementation for compiler support hosted on UTS on an IBM 4341. The current status of this project is not known.

e. An Operational Definition of the CAIS is being funded by the Ada Joint Program Office to Dr. Timothy Lindquist. This was initially started at Virginia Polytechnic Institute and is currently being continued at Arizona State University. The full CAIS operational definition will be completed in early 1987.

f. The government is funding a VHSIC Hardware Description Language (VHDL) effort which is a partial implementation for database support; this was in implementation stage in mid-1985.

g. Texas Instruments completed a partial implementation to support their APSE Interactive Monitor (AIM) tool. This was hosted on the Data General Ada Development Environment and was rehosted to the Digital Equipment Corporation's VAX running VMS.

h. The University of Houston at Clear Lake did a partial implemenation of the CAIS as a class project under Dr. Charles McKay on the Data General Ada Development Environment. This work was completed in the spring of 1985.

i. Honeywell completed a partial implementation to exercise their compiler in 1985.

j. It is expected that the KIT support contractor (beginning their KIT support contract in January 1986) would do a prototype implementation of the CAIS.

k. The CAIS 2.0 contractor will be developing a full implementation of the CAIS.

l. The government-sponsored WIS effort had been working on a "parallel" implementation for its SDME. Little is known at this time on this effort.

At the time of the Workshop, there were four CAIS implementation efforts in progress.

Dr. Timothy Lindquist has been developing an Operational Definition of the CAIS, first at Virginia Tech, and currently at Arizona State University [3].

The MITRE Corporation [6] and GOULD, Inc. [5] have been developing prototype implementations of the CAIS.

At this Workshop, it was announced that IBM has begun a prototype implementation of the CAIS; no further details were given.

Since the Workshop, it has been announced that a team consisting of SofTech and Compusec has been awarded a contract to design the next version of the CAIS. As part of this contract, there is a requirement to develop a prototype for the proposed MIL-STD-CAIS as well as for the CAIS 2.0 design.

The current contract that has been awarded to TRW, Inc. for KIT support also includes a provision for a CAIS prototype.

1.2 Goals of the Workshop

The first organized meeting of the designers of the CAIS and several implementors was held October 29th and 30th, 1985 at the University Inn in Pittsburgh, Pennsylvania. A list of attendees is given in the appendices. The goals of this first meeting were:

1. The exchange of information between the designers of the CAIS (members of the CAIS Working Group (CAISWG) of the KIT/KITIA) and those groups (mainly represented by the CAIS Implementors Working Group (CIG) of SIGAda) actually doing prototype or development of the CAIS, resulting in a set of issues that need to be addressed and possibly acted upon during the evolution of the proposed MIL-STD-CAIS.

2. The introduction of other environment ideas (most notably UNIX and PCTE) to the CAISWG and CIG for possible inclusion into the CAIS during its evolution.

3. The discussion of these issues as well as ones to be raised at the meeting.

1.3 Participants

Besides members of the CAISWG and several implementors, several representatives from the Software Engineering Institute (SEI) and two members of the WIS Operating System Task Force were present. Four members of the research staff of the Institute for Defense Analyses (IDA), the host for this meeting, also attended.

Members of the CAISWG included: Herman Fischer, chair of the KITIA, who also has been extensively following the European environment efforts; Tim Harrison of Texas Instruments, who was one of the original members and who has provided most of the inputs to the I/O section of the document; Tim Lindquist of Arizona State University, who has been the leader of the CAIS Operational Definition effort; Hans Mumm, acting chair of the KIT; LCDR Philip

Myers, who is currently the Navy representative on the AJPO and who has been foremost in the security aspects of the CAIS; Erhard Ploedereder of Tartan Laboratories, who has provided much input to the data administration facilities and the node model of the CAIS; Carl Schmiedekamp of Naval Air Development Center; Ray Syzmanski, who is the chair of the Evaluation and Validation Team (sponsored by the Ada Joint Program Office); and Richard Thall, who is the chief architect of the Army's Ada Language System.

The implementors were represented by Gould, MITRE, and IBM. GOULD was represented by Pete Carr, Mike Doub (Manager, Ada Project Development), and Robert Stevenson (Manager, CAIS Project). Representing MITRE at the Workshop were Helen Gill, Chuck Howell, Robbie Hutchison, Mike McClimens, and Tana Reagan. IBM was represented by Jeff Vermette of Federal Systems Division in Manassas, Virginia.

Two members of the WIS Operating System Task Force were also present. Jack Stankovic, currently at Carnegie-Mellon's Department of Commputer Science is the chair of this task force. Also in attendance was Mike Liu of Ohio State University.

The chair of the CAIS Working Group of KIT/KITIA is Jack Kramer. He has been the Navy representative of the AJPO and one of the original designers of the CAIS. He is presently Director of the Computer and Software Engineering Division (CSED) at IDA.

1.4 Format of the Workshop

The agenda is included in the appendices.

General areas of discussion included: CAIS Security requirements, the CAIS and other environment development efforts (including PCTE, the WIS Operating System, UNIX, etc.), Input/Output packages in the CAIS, the CAIS package LIST_UTILITIES, Distributed CAIS, CAIS and the Space Station, and a list of issues brought forth by the implementors.

After the welcome was given by Clyde Roby of IDA, general introductions were made. Information was then imparted concerning the status of various CAIS-related activities. Following this, discussions then began on some of the major issues.

2.0 Status of CAIS-Related Activities

Hans Mumm, acting chairperson of the KIT (in Patricia Oberndorf's absence), stated that there was currently no KIT support contractor on board (the KIT support contract was awarded to TRW in December 1985). Neither was there a contractor announced for the CAIS 2.0 work (the team of SofTech and Compusec was awarded this contract in December 1985).

There was some concern about the status of the CAIS comments that have been submitted by the CAIS Implementors and reviewers of the CAIS document. Comments should still be submitted to CAIS-COMMENTS and the KIT support contractor will address these soon.

The major differences between the proposed MIL-STD-CAIS and CAIS 2.0 is that there will be a resolution of differences between the proposed MIL-STD-CAIS and the Requirements for CAIS 2.0. Any technical changes will first be brought to (Hans Mumm and) Patricia Oberndorf as chair of the KIT and these will then be brought before the KIT/KITIA as appropriate.

The CAIS 2.0 contractor will be developing a prototype implementation. It was emphatically stated that there was a requirement for upward compatibility of CAIS 2.0 with the proposed MIL-STD-CAIS. In the standardization process of the proposed MIL-STD-CAIS, there should not be any major differences as CAIS 2.0 evolves.

There are two major reasons for the standardization of the CAIS:

1. Getting the attention of all the necessary (environment and Ada) communities
2. Opening up the CAIS for examination by a wider audience

People are beginning to sign up for the standard. It will have prototype implementations and will be examined in other means, too. As long as the standard becomes a product, industry should accept it.

Some concern was raised about the perceived difference between the proposed MIL-STD-CAIS (CAIS 1.x) and CAIS 2.0 because of the number. It was brought out that the CAIS 2.0 contractor would only slightly evolve the CAIS similar to the way that the Ada language standard MIL-STD-1815 evolved to MIL-STD-1815A. It was also suggested that a different means of referring to CAIS 2.0 would be appropriate at this time because of industry perceptions.

3.0 Security Issues

The first major issue addressed was that of security. Can the CAIS Security Model be implemented on an unsecure host? CAISWG's intent was to design the CAIS such that it could be implemented on a secure kernel. An implementation can make it better but it still is not really secure.

MITRE deferred their implementation of any Security Model because most tools that they have implemented do not call on the security interfaces. Security people within MITRE have questioned the implementation of the CAIS on top of a secure operating system and suggested that it must be re-evaluated in this context. MITRE was also concerned with the history of the Security section of the CAIS being subject to change the most in the near future – they are basically waiting until that portion of the CAIS stabilizes.

LCDR Philip Myers, Navy representative of the AJPO and also representing the security community, was disturbed that none of the implementators were evaluating the CAIS Security Model to determine if the CAIS can be implemented on a secure system.

Both MITRE and GOULD are currently putting strictly application code on top of the CAIS; GOULD did address the discretionary access controls.

It was suggested that if the implementation is done this way, then the whole purpose of a secure system is defeated. The reply to that was "Not really – all users coming into this system would be on the same security level."

The only requirement concerning security in the CAIS is that the CAIS functionality not get in the way of security requirements mentioned in the so-called "Orange Book", Trusted Computer Security Evaluation Criteria [12].

There are two issues if one settles on a Security Model similar to that of the proposed MIL-STD-CAIS:

1. For a CAIS implementation on a Multi-Level Secure system, does it have problems in the Security Model?

2. For a CAIS implementation on a System Level High system, mandatory access control must be the default.

The CAISWG is interested only in the semantics of the CAIS. This cannot go counter to security.

LCDR Myers said that it is necessary that the CAIS does indeed specify default situations, especially when the CAIS is booted up on an unsecure system or on one that has no underlying security mechanisms. In the design of the kernel, the CAIS implementation must be addressed in order to determine that it does not belie security.

Is the CAIS giving a false promise concerning security? Some of the security portions need to be pushed into the kernel. There needs to be a separation of the secure mechanism in the kernel from the CAIS.

It was brought out that an implementation on top of a secure kernel is not really a problem until the database is moved to another implementation. The CAIS 2.0 contractor will have to address the movement of tools and data between secure systems.

Tim Lindquist continued the discussion by raising the following three questions:

1. Can the CAIS be implemented on a bare system and be secure?
2. Can the CAIS be implemented on a secure system and the result be secure?
3. Can the CAIS provide the appropriate functionality for tools on a secure system?

He said that the Workshop really needs to address the first two questions.

With respect to the "Orange Book", an evaluation needs to be done of both the underlying operating system and of the CAIS. The evaluator does not care where the Trusted Computer Base is located.

The design of the interfaces for a bare CAIS implementation is different from the design of the interfaces for a piggyback implementation. The answer to (1) above is yes. There seemed to be no general answer to (2) above.

The CAISWG is trying to design the CAIS so that (2) above can be answered positively, i.e., so that an implementation of the CAIS on a secure kernel can be done with the result being secure. If the CAIS cannot be implemented on a secure kernel, then a major problem definitely exists.

The CAIS does not make additional promises. The CAIS can be implemented on a secure system but still be unsecure (this is a poor implementation, though).

The CAIS does not have a Mandatory Secure Model and any Mandatory Secure Model can be inserted in the CAIS; (1) above must be addressed.

Can the CAIS be implemented on top of a Trusted Computer Base? The only example of a secure system that we currently have is MULTICS, but it does not have any Ada language support tools on it.

Richard Thall of SofTech noticed that there is a division of labor between the CAISWG and the CAIS Implementors Working Group. The CAISWG developed the interfaces. The implementation of the interfaces will have to be devloped by a lot of different groups in a mono-level (system high) mode and then moved (for maintenance) to a Multi-Level Security sytem. He said that items should be marked in the proper level so that when that movement occurs, it would go to the proper place in the Multi-Level Secure system. He asked the implementors, "Can a marking system be developed to aid this?"

LCDR Myers indicated that these labels are useful in mono-level systems but there is a danger that the people who use them would "lull" themselves over time to trusting the multi-level labels in the mono-level system.

Since this meeting, further discussions have continued. There is a definite requirement for security in the Requirements and Design Criteria document for the CAIS [8]. However, nothing at this time has been firmed up.

4.0 General Package Structure

The general package structure of the CAIS leaves a lot to be desired in the minds of many of those that were present.

Tim Lindquist, who has been working on an Operational Definition of the CAIS, said that the usage of type NODE_TYPE (a limited private type) in the CAIS needs to be reviewed.

It was generally agreed by the CAISWG membership present that rearranging some of the CAIS packages was all right for an implementor to do – especially the NODE_DEFINITION package. Most implementors have moved all type definitions to the CAIS package level, i.e., to the outermost package level.

Concerning an implementation of package ATTRIBUTES, Tim Lindquist indicated that an attribute could have complete access to LIST_TYPE. The way it is, attributes can be implemented using the LIST_UTILITIES package, or, by converting them to an object of type STRING and then into an internal representation.

Erhard Ploedereder, one of the original designers of the CAIS, said that this is the same problem as with type NODE_TYPE. All limited private types should be accessible in CAIS implementations.

It was noted that process control (package PROCESS_DEFINITIONS) uses LIST_TYPE for its results list. RESULTS_LIST semantics states that RESULTS_LIST can't be seen other than through the interfaces (i.e., not through any of the ATTRIBUTES packages).

The idea behind having specific interfaces for some of the attributes was that an implementation would get the information from the operating system rather than from attributes on the node.

Jack Kramer, chair of the CAISWG, said that all limited private types should be moved to the (outermost) CAIS package. He said that in the design of the CAIS, there was a desire not to limit how intelligent implementations should be.

4.1 Exceptions

Another general problem concerned the definition of exceptions and what they actually cover. MITRE has encountered this problem and has suggested an alternative. Following is the discussion on exceptions, exception names, and the possibility of creating a higher level package containing just the CAIS exceptions.

There are multiple and overlaying exceptions. Some have broad reasons for exceptions (USE_ERROR, for example). There are minor semantics with respect to some of the cases but clarification would help the tool writer.

When asked, "Does this proliferate the number of exception handlers?", MITRE's response was "Not really. It does not change the semantics of exceptions."

MITRE identified two different problems:

1. USE_ERROR is defined in two different packages (there should be just one at the CAIS package level). This would allow us to continue to follow the Ada language example of renaming exceptions.

2. There is a need to expand the exceptions to cover the real reasons for their occurrence. MITRE has 18 exceptions defined and they are defined in one place.

The implementors said that the current CAIS is not really granular enough to explain the reasons why exceptions occur.

MITRE's suggested CAIS_EXCEPTIONS package is shown on the next page.

```
-- MITRE's suggested package for exceptions in the CAIS.

package CAIS_Exceptions is

        Node_Name_Error                 : exception;
        Node_Use_Error                  : exception;
        Node_Status_Error               : exception;
        Node_Intent_Error               : exception;
        Node_Lock_Error                 : exception;
        Node_Access_Error               : exception;
        Node_Security_Error             : exception;
        Node_Mode_Error                 : exception;
        Node_Data_Error                 : exception;
        Node_End_Error                  : exception;
        Node_Device_Error               : exception;
        Node_Layout_Error               : exception;
        Pathname_Syntax_Error           : exception;
        Existing_Relationship_Error     : exception;
        No_Such_Relationship_Error      : exception;
        Predefined_Relation_Error       : exception;
        Predefined_Attribute_Error      : exception;
        List_Syntax_Error               : exception;
        List_Use_Error                  : exception;
        List_Search_Error               : exception;
        List_Name_Error                 : exception;
        CAIS_Internals_Error            : exception;
        Host_Error                      : exception;

end CAIS_Exceptions;
```

4.2 Interaction among CAIS Input/Output Packages

The combination of Input/Output packages in the CAIS opened up several questions with respect to interaction of different packages on the same file or node. This was the next subject of discussion.

CAISWG: Interaction of different I/O packages on the same file was specifically left out of the CAIS document.

CAIS Implementors:
Did you explicitly go away from any model?

CAISWG: The idea of passing a file to another process (since the process does not know what kind of file, e.g., disk, terminal, etc., is passed to the other process). Do we not allow passing of file nodes?

CAIS Implementors:
Separate file types would adhere to the Ada model.

CAISWG: It has been suggested that we do away with file types, particularly due to access control.

Access synchronization is good for the nodes. Historically, we wanted to stay compatible with the Ada I/O packages, but it looks like we'll go with the "get rid of the FILE_TYPEs" suggestions.

CAIS Implementors:
That would seem to be nice and consistent.

CAISWG: We've tried to keep the world as uniform as possible.

CAIS Implementors:
It would help to have interactions or the lack thereof explicitly stated in the CAIS document. Basically, we are arguing for clearer semantics.

It must be made clear what happens on mixed file types.

CAISWG: The level lost here is less consequential. We gain advantages as tools become more portable. Although there might be 1-2% degradation, the tools would be more useable. UNIX offers everything as a file, whether disk, terminal, or whatever. We need that uniformity.

CAIS Implementors:
There are two philosophies: one universal type versus many types but maintaining uniformity.

I/O redirection is necessary.

CAISWG: In designing the CAIS, who is it that manipulates the mapping between the physical and the logical mappings of the world? We want to provide I/O redirection where the tool doesn't care. We want the runtime flexibility of I/O reassignment.

CAIS Implementors:
Things need to be nailed down more explicitly in the CAIS document.

CAISWG: We did not want to duplicate the Ada LRM's Chapter 14 semantics. Implementation-wise, file types are usually different.

CAIS Implementors:
TEXT_IO and SCROLL_IO really don't mix.

Intermixing of them is a problem (keeping track of the cursor, etc.).

CAISWG: In the design of the CAIS, numerous discussions were held about tools and packages. When a tool uses the lower level of abstraction supported by the device, it should be allowed to do that. On the other hand, it is unreasonable to expect that a tool using a more sophisticated device to behave properly on a less capable device. In a sense, we want to achieve portability upwards.

CAIS Implementors:
We need to nail down these interactions.

CAISWG: A tool should not know where its I/O goes to.

CAIS Implementors:
Redirection is different from access mechanisms.

Redirection is necessary; interaction is not desired.

CAISWG: We must properly define semantics for all packages and interfaces.

4.3 Processes

The discussions concerning Input/Output led into discussions about the spawning of processes and into the discussions dealing with logon/logoff and "dead" processes.

CAISWG: For the interface CREATE_JOB, the process execution tree remains as long as the process tree remains. The background job can be created on another process tree. Then you can logoff and let the process run.

CAIS Implementors:
In the daily use of UNIX, one shifts things in and out of the background.

When one logs off, the active processes go up a level to the subinitiator.

CAISWG: UNIX is the only operating system that provides it.

There should be no overhead in keeping "dead" nodes in the tree around.

CAIS Implementors:
This is an undefined example of interaction between processes and file nodes.

CAISWG: Once you get to processes, you have a problem.

Logout and login are not defined in the current CAIS. There are several implementations possible.

We still have the ownership problem of the terminal node.

That is still an implementation issue; one could kill the process.

CAIS Implementors:
Implementation issues need to be explicit in the CAIS document.

CAISWG: Then we must define logout, login, the shell, etc., but "Implementors have favorite biases and we go towards a higher calling."

5.0 CAIS Iterators

CAIS iterators evidently caused some problems for the implementors.

Herman Fischer, chair of the KITIA, has been following the European efforts on environments very closely – in particular, he has been extensively involved in following the Portable Common Tool Environment (PCTE) effort. He said that the PCTE iteration paradigm returns tables corresponding to a group of names.

Mike McClimens of MITRE indicated that:

1. The operation set seems fairly small – it is only done in alphabetical order. There might be room to add more iterators. It would also be good to return the number in the set.

2. For node iterators, what is really in the node set? A lot can happen by the time a node is put into the set and by the time it gets used.

Erhard Ploedereder said that it's possible that the set doesn't have to be created. An implementation could simply use a pointer (and thus do complicated things at the iterator). The same attitudes are present here as with the operating system – tough luck if something unexpected happens.

Richard Thall, chief architect of the Army's Ada Language System (ALS), said that the ALS has added interfaces to do work in blocks or groups. The invocation of the CAIS interface usually invokes an operating system interface or supervisor call interface so reducing the number of calls is beneficial. Beyond the bounds of the CAIS, the structure of iterators is similar to wildcarding. This is not too well defined in the ALS. The CAIS idea of a node name is not well developed. Wildcarding over attribute values (presence of attributes versus contents of attributes) is important in structuring iterators. The tendency should be in getting sets, groups, or blocks.

Herman Fischer indicated that it would be useful to return a block to an Ada language interface with work being done in routines in between. If one does a string match and a count is requested, one needs this. An open handle to a node is needed. The primary relations are not affected. In a way, this is a CAIS issue, but the user may want the information but would have to go through it.

Tim Lindquist said that this is an issue of late binding versus early binding.

Even late binding has surprises. Something could be inserted as tree traversal is being done during the processing of the iterator. Nothing can be done about safety with respect to the interfaces. The only other issue is efficiency. The option is open to the implementors.

If the names are returned to a user *en masse*, the user has it. Otherwise, different things could happen in different cases.

When an open node from an iterator is passed to the PATH_KEY or PATH_RELATION function, it will return a relationship that that came across. Delayed binding implies going after it by the primary key. Whether the relationship exists or not, the string is returned.

Going a level down, some boundary conditions are undefined. More explicitly defined semantics are needed. For example, what does "iterator being exhausted" mean in the GET_NEXT procedure?

MITRE talked about ways of picking things (in their development effort) – currently there is PRIMARY_ONLY, etc. More could be added, e.g., SECONDARY_ONLY, PRIMARY_AND_SECONDARY, etc. They would also like a way to get only file nodes or only process nodes, etc.

The solution is to have an enumeration type.

Jack Kramer replied that this is where real tools on a CAIS implementation is useful. If a lot of tools use something, then CAISWG would pay more attention to the implementors in that particular area.

MITRE uses a flat file structure; they don't map to the UNIX directory structure. Instead of returning open node handles, it would be better to return the string name. For the most part, something needs to be done so the node handle is returned instead of the string. This was the argument.

Then AND instead of OR is needed for returning the string name.

Although some members of the Workshop would rather see regular expressions since user's tools really could do something with regular expressions, it was pointed out that regular expressions may not be needed when attributes are present in the CAIS.

One is constantly comparing trees when two baselines need to be compared. In the ALS, it is done in alphabetical order and that will be the standard. Richard Thall told the Workshop that an ALS user called (the ALS Hotline) not too long ago and wanted a different ordering. No matter what order is picked, someone will differ.

A path iterator and a comparator could be used. One can do it with existing interfaces if it is built on top of them.

Erhard Ploedereder gave a little discourse of the background for having the KIND attribute present in the CAIS. Different kinds of nodes will probably be implemented differently. It's an implementation concern as well as a usage concern.

Everyone should look at the need for additional interfaces. There may be a need for more specific ones.

6.0 WIS Operating System (WOS) and the CAIS

6.1 Introduction to the WIS Operating System

The World Wide Military Command and Control System (WWMCCS) is an arrangement of personnel, equipment (including automatic data processing (ADP) equipment and software), communications, facilities, and procedures employed in planning, directing, coordinating, and controlling the operational activities of U.S. Military forces.

The WWMCCS Information System (WIS) is responsible for the modernization of WWMCCS ADP system capabilities, including information reporting systems, procedures, databases and files, terminals and displays, communications (or communications interfaces), and ADP hardware and software. The WIS environment is a complex one consisting of many local area networks connected via long distance networks. The networks will contain a wide variety or hardware and software and will continue to evolve over many years.

The main functional requirements for WIS are presented in [13]. Briefly, the functional requirements have been categorized into 7 areas.

1. Threat identification and assessment functions involve identifying and describing threats to U.S. interests.

2. Resource allocation capabilities must be provided at the national, theater, and supporting levels.

3. Aggregate planning capabilities must provide improved capabilities for developing suitable and feasible courses of action based on aggregated or summary information.

4. Detailed planning capabilities must provide improved methods for designating specific units and associated sustainment requirements in operating plans and for detailing the sustainment requirements in supporting plans.

5. Capabilities must be provided to determine readiness, for directing mobilization, deployment and sustainment at the JCS and supported command levels, and for promulgating and reporting execution and operation orders.

6. Monitoring capabilities of the system must provide the information needed to relate political-military situations to national security objectives and to the status of intelligence, operations, logistics, manpower, and C3 situations.

7. Simulation and analysis capabilities must include improved versions of deterministic models that are comparable to those contained in the WWMCCS.

In order to support these high-level objectives, the WIS system software must provide an

efficient, extensible and reliable base upon which to build this functionality. To develop such system software several projects are planned for prototype foundation technologies for WIS using the Ada programming language. The purpose for developing these prototypes is to produce software components that:

1. Demonstrate the functionality required by WIS,
2. Use the Ada programming language to provide maximum possible portability, reliability, and maintainability consistent with efficient operation, and
3. Demonstrate consistency with current and "in-progress" software standards.

Foundation areas in which prototypes will be developed include:

1. Command Languages
2. Software Design and Analysis Tools
3. Text Processing
4. Database Tools
5. Operating Systems
6. Planning and Optimization Tools
7. Graphics
8. Network Protocols

The most important ingredient for a successful WIS is the design and implementation of a suitable distributed operating system. The WIS Operating System (WOS) is a distributed operating system in the sense that it provides an abstraction of a single system across network connected multiple machines. The design of the WIS Operating System is a well-balanced design that has significant potential for meeting the requirements of WIS. For example, effective performance is achieved by providing a minimal kernel that optimizes Local Area Network (LAN) Inter-Process Communications (IPC), contains a very fast context switch and supports "lightweight" kernel tasks. Security is supported in the kernel by having clearly delineated address spaces, basic mandatory access control and all communication controlled via the IPC mechanism which can ensure that the proper security access is followed. Security is also supported outside the kernel by (1) "alias" processes which implement and serve as safeguards for inter-cluster communications, and by (2) an authentication agent. Fault tolerance is provided, in part, by the distributed nature of the system, as well as by the fault tolerant distributed file system. Extensibility is enhanced because of the multilevel and modular design of the WOS as well as the use of the Ada programming language and adherence to the CAIS.

The structure and modularity of the WOS can be explained by considering three main levels together with the concept of an agent. An agent is a module that implements one or more Ada packages to provide some service such as authentication, logging and auditing, or secondary storage management. The three levels are:

Level 1 (Kernel): The kernel provides an efficient base for transparent (network-wide) IPC, security, and basic process, main memory and device support. It also provides basic support for the CAIS and the concept of an object. Each type of object is viewed as an abstract data type. WIS Operating System objects include open files, atomic transactions, jobs, processes and virtual spaces. The kernel provides operations for invoking operations on objects between processes, such as found in the message passing scheme. It also provides operations for changing the amount of valid memory associated with a task plus mapping portions of files in and out of the address space. Other types of objects and operations on them can be defined at the other levels. Further, language processors are free to define other types of objects either using these basic WOS objects or independently.

Level 2 (Run-Time Support):
Facilities that need not be implemented in the kernel are implemented in server processes that execute outside the kernel as well as by so-called run-time procedures that execute in the address space of the invoker. The run-time support level provides this necessary Operating System functionality that is not included in the kernel. This level is extensible and initially includes the following agents:

1. Secondary Storage Memory Management
2. Program Execution Module
3. Authentication
4. Time Synchronization
5. Command Language Interpreter
6. IPC support
7. Logging and Auditing
8. Alias processes support
9. Print Server
10. Multi-window server
11. I/O drivers
12. Transaction manager
13. Name Server

Level 3 (Application):
The application level includes application programs and other agents which are not necessary for run-time support. This includes the DBMS, user application tasks, and non-essential Operating System utilities.

It is intended that most of the WOS will be implemented in the Ada programming language and adhere to the CAIS.

6.2 Overview of the WIS Operating System (WOS)

Jack Stankovic spoke on what the WOS Task Force has been doing; he is the leader of the WOS Task Force. The following paragraphs summarize his discussion.

Most things that we want to do in a distributed operating system can be tailored to the CAIS directly, or hidden in the kernel (below the CAIS interface).

There is one scheduling policy for local tasks and a different scheduling policy for distributed tasks. The WOS is designed in about 13 different major packages – these form the basis for distributed systems.

Jack Stankovic was concerned with pushing off issues of distribution, security, and tool interoperability by the CAISWG. WIS is largely a DBMS. The old WWMCCS is commercially oriented, with COBOL as the main implementation language.

The purpose of WIS is to provide information. It is also a development system. The CAIS model is also applicable for mission critical systems and for mission support. The CAIS has no realtime process control, but there are no requirements for WIS in this area at this time.

How does this compare with the WIS Software Development and Maintenance Environment (SDME) effort? The WIS Task Force is funded by the Joint Program Management Office. This program is to look into risk reduction technology and development. The SDME is run by the Special Projects Office out of Bedford, Massachusettes. Hopefully, the WOS will be implemented.

Each of the four members of the Task Force has given about two-to-three days per month effort towards this work. The purpose for representation of the WIS Operating System Task Force at this meeting was to see where problems have surfaced in the CAIS. All the issues here discussed are relevant.

The kernel is influenced by the CAIS, but does not implement the CAIS.

A lot of the CAIS interfaces are implemented by the first two levels of the WOS Kernel – primarily files and processes, so there will be additional software built on top of it.

The CAIS could be implemented on top of the WOS. Basically, we are doing a CAIS implementation on a bare machine. There is a quite detailed specification for the WIS

Operating System [13].

How much distribution should be viewable by the user? How much distribution should be viewable by the tools? The tool being able to specify a particular host may be bad, but specifying parallelism should be present in the CAIS.

Ada/CAIS parallel specifications issues have arised. Do you allow a task to migrate after execution has begun? Our answer is no.

6.3 General Discussion

The following general discussion is in the form of comments and questions given by individuals and Jack Stankovic's responses to them.

Herman Fischer: How do you determine where a task goes?

J. S.: A process can be distributed. The scheduling algorithm needs a lot of state information. We don't want a localized scheduler policy. Localized and globalized schedulers exist.

Jack Kramer: I define a "system" as being the runtime system, whether that is on a single processor or is distributed over several processors. One runtime system will worry about the distribution of processes.

J. S.: The PCTE wants to force the initial execution of a certain task to work on a certain processor. There are a lot of trade-offs involved.

Chuck Howell: Is the model extended to heterogenous hosts?

J. S.: As long as there is an adherence to protocols, there is no problem. We won't [currently] move tools after execution begins. However, if the [WOS] contractor wanted to propose that, they would have to come back to the Task Force to have their design checked out. There are systems today that move processes from host to host on the fly.

Fault tolerant mechanisms are separate from the initial process distribution. WIS requirements have kept the complexity down. There are other ways of checking this out, e.g., N processes communicating loosely so that N-1 continue to execute when one goes down. Dynamic load balancing may not be necessary.

Richard Thall: What taxonomy of task organization/distribution is used?

J. S.: A local node could be a multi-processor with shared memory. Most multi-processors are similar. There are some mechanisms for hiding distribution. We do have LAN protocols and gateways.

The CAIS is going to be used in a distributed manner. But there is a problem in defining the CAIS before some of these uses, i.e., before complicated tools become available.

Jack Kramer: That was one of the issues CAISWG had a horrible time with. At the CAIS level, we are talking process to process. We looked at the initial Ada tasking model but dropped back to the current queue model.

Herman Fischer: Did you talk about a security model?

J. S.: We have discretionary access control. We are to have mandatory access control but are interfacing with the Database Group on this. The granularity of data elements with respect to access control is still being addressed. We need some kind of mechanism. The Database Group wants the operating system to support it; the Operating System Group wants the Database to support it. We are going for a B3 level on the host node level. Distributed nodes are B3 level on secure LANs communicating via encryption methods.

Jack Kramer: The National Computer Security Center is evaluating systems against the evolution of the Orange Book. They are currently generating Network Evaluation Criteria. They also will generate Database Evaluation Criteria. They are a long way aways where Network Criteria is generated to be accepted by the security community.

Herman Fischer: There will be a problem qualifying these things over several links.

LCDR Philip Myers:
The view of it now is to view the network as a system and evaluate it as a system. WIS is not going into this area blindly.

Jack Kramer: This is the foundation technology part of WIS. We may not get there. It might be too expensive. But we have the ability to ask to try to get there.

J. S.: We are trying to require it – various aspects of the B3 model. The CAIS keeps information about processes. In a distributed environment, we need additional state information that can be added with attributes.

Jack Kramer: At one time, CAISWG had defined a layered set of process information but realized that we were too close to the runtime system. What kinds of things could be useful – that can be added as attributes?

This last question led into a discussion of attributes, particularly time-related attributes.

6.4 Timestamps and TIME attributes

Timestamps and other related attributes were found useful by MITRE, especially dealing with configuration management and configuration control tools.

MITRE indicated initially one can provide a partial ordering across the distributed environment. This is still useful. They still want to deal with the time problem up front. They are concerned with usability of tools.

CAISWG stayed away from "timestamp" because it was "undefined".

The WOS Task Force is still cautious with respect to timestamps. They are not using concurrency control with timestamps. They don't want a lot of overhead, but must put in the extra cost if needed.

Everyone agreed that there is a need for an attribute called TIME to solve these problems.

Jack Kramer indicated that the node model is in the CAIS because of configuration mangagement issues. If TIME is something needed by a lot of tools, it makes sense to put it into the CAIS in a standard way. CAISWG had the feeling that there were many TIMEs (created, accessed, etc.). If the concensus around is about what TIMEs are needed, then CAISWG should add them in.

MITRE asked, "Without TIME, how is compilation order enforced? All Ada compilers currently support it via TIME."

The CAIS implementors should decide which attributes are needed and bring their suggestions to the CAIS designers. Then when there is agreement, the CAIS designers will put it in the CAIS. The implementors should determine which TIME-related attributes to have and agree on them. CAISWG didn't know (from the infinite choices) what to do with respect to attributes in general. The implementors should come to an agreement of what attributes they want and the CAIS designers will put them in. GOULD indicated that there are already START_TIME and FINISH_TIME attributes on process nodes.

As soon as one goes to a binary representation of date, there is a problem. There should be a standard string representation or something else for the external representation. The addition of predefined attributes and relationships should be up to the implementors and the user groups.

Jack Kramer indicated that the CAIS designers are not in a position to know the complete set of predefined attributes and relations. Implementors are in a better position to know that. It was brought out that all of the members of the CAISWG had their own favorite attributes and relations, rather than no idea of what should be in the CAIS.

6.5 Scheduling for Distributed Systems

Jack Stankovic discussed scheduling in the WIS Operating System.

The scheduling algorithm for distributed systems needs more information. This was kept outside the CAIS – in a package STATISTICS in the WIS Operating System design. There are no group related process things in the CAIS.

Concerning the question (in the original issues list handed out at the beginning of the Workshop) about secondary relationships, we are concerned about maintaining primary and secondary relationships consistently across a distributed environment. Should the pair of relationships be maintained, especially if it is moved across the distributed environment?

The PCTE has all of the bidirectional relationships: one-to-one, one-to-many, and one-to-none. However, the PCTE has a major security problem. Also, the PCTE does not have the notion of a primary path.

The other issue concerns deletion of secondary relationships. Garbage is left around. Would the Entity-Relationship model require deletion of relationships when the node is deleted? No, the CAIS has no way of accessing those relationships once the node goes away.

6.6 Distribution and the Space Station

There was some discussion about the Space Station program and its use of a distributed CAIS. [10]

Quoting from [10], the concerns that were brought forth in this article included the deferment of the following areas:

1. Distributed environments. . . . Currently deferred is the decision whether or not to provide to the user explicit CAIS interfaces to control the distribution of the environment, including designation of where nodes exist and where execution takes place. Note that a set of distributed processors could include one or more target machines."

2. Inter-tool interfaces. The current CAIS does not define . . ., the command processor language syntax, . . . or the interaction between the runtime system and debugger tools. Currently deferred are decisions regarding . . . and whether or not to place constraints on the runtime system to provide process execution information."

3. Interoperability. The current CAIS . . . does not define external representations of data for transfer between environments or between a host and a target."

In the process of evolving later versions to meet the needs of distributed applications, it will be necessary to decompose the process node of the current version to support a finer grain of program element representation. The process node in conjunction with a queue node provides a meta level representation of all program units of an Ada program. If the program to be executed will spawn three tasks (each to be executed simultaneously on a different processor which is remote from the processor responsible for the parent process) the currently proposed process node (with its associated queue node) would be the single required CAIS repository of information on the state of the program, resources being utilized, etc. It seems obvious that for purposes of debugging, performance analysis, dynamic reconfiguration support, and many other features that characterize distributed applications, a finer grain of representation is needed.

7.0 IMPORT/EXPORT and the CAIS

There was a discussion concerning the use of data and tools moving back and forth from the CAIS environment to the operating system to which the CAIS interfaces. Discussions were about bringing data files inside the CAIS environment and then back out again (especially for Ada source compilation). A related topic was the usage of tools that normally run outside of the CAIS but that could run inside the CAIS using CAIS objects for input/output.

The discussion began with the question: Can MITRE port GOULD's database to MITRE's database?

It would be a good exercise to define the CAIS node attributes and relationships using an external representation. An import/export of the node structure itself would also be useful. Issues of archiving need to be addressed. Solutions could be enumerated.

The ALS has a tool to collapse the tree and output it to a tape. The ALS contractors are trying to come up with a bridge of the ALS to the CAIS.

Whether something like that should be in the CAIS or not is debatable. Implementors could come up with a tool.

The external form could be standardized.

CAIS 2.0 has a requirement for an external form.

Archiving, backups, and restoration highlight the need for special interfaces in the CAIS to bypass configuration management in order to restore old versions.

8.0 LIST_UTILITIES

Mike McClimens of MITRE led the discussion on the LIST_UTILITIES package.

This is a very cumbersome set of interfaces. It provides the proper services but in a clumsy way. We will present a proposal that will not change the utilities but make it more useful.

Jack Kramer indicated that this was effectively ignored in the first three versions of the CAIS. It was revised in a major way in the last month of the design of the proposed MIL-STD-CAIS. Most of these came out of TRW's implementation.

Tim Lindquist said that the current model is difficult to understand. Why are there so many interfaces?

It seems to MITRE that reasons for going from a string representation to a type representation was for efficiency of implementation.

In terms of user convenience, it might be efficient. CAISWG has been constantly importuned because attributes do not have types. In this sense, the LIST_UTILITIES model is a response to that. There used to be a FIXED_POINT package as well; this was eliminated. The same problem also exists in package DIRECT_IO. CAISWG talked about changing token information to indicate what type the value was.

The main reason for introducing tokens into the CAIS was for access control, instead of dealing with them in quoted form. For predefined attributes, that type support is provided in interfaces in an enumeration type.

CAISWG wanted to use the Ada aggregate for lists. It was suggested in the wording to the effect that it should look like an aggregate. It should guarantee that the string going in and coming out should be the same after the double conversion. Some examples should at least be cited, e.g., page 193 (d) and maybe page 213 (5.4.1.23) of the CAIS document [2].

It was noted that GOULD did not have problems in this area.

Mike McClimens of MITRE continued. We have proposed an expanded token model to replace the current model. Lists are made up of tokens. The TO_TOKEN operation is used

for each type to put into the list. There is only one EXTRACT interface; it extracts a token. It really does make a reduction in the overall number of interfaces. Also, currently REPLACE is used for replacing type by type; we are proposing that REPLACE can replace token by token.

Herman Fischer asked, "Doesn't this take away from the UNITY of the current interfaces?"

Richard Thall indicated that if it is in the user's domain, that could invalidate the lists.

Tim Harrison said that the user could, via UNCHECKED_CONVERSION, invalidate it anyway.

It was generally agreed upon that MITRE's suggested LIST_UTILITIES package was a cleaner model (see the appendices).

Mike McClimens continued. Most things really stay the same. The separate packages disappear. All the list item stuff should be in one package, anyway. About the only other things that happen is the KIND operation which operates on a token. KIND could be user extensible. A possible problem that one runs into, however, is the FLOAT and the FIXED_POINT problem.

Another problem that was identified is that MITRE has a TO_LIST operation that becomes difficult for extensibility. If one is going to want a general tool to work, it's not too terrible to use the predefined string, or even ASCII.

9.0 Exceptions in LIST_UTILITIES

MITRE also had specific suggestions for exceptions in LIST_UTILITIES. They suggested moving exceptions out to a common layer and that the number of exceptions be expanded. MITRE's suggested CAIS_Exceptions package is contained elsewhere in these proceedings. Currently, USE_ERROR is like a garbage can, i.e., it is a general error collector.

Erhard Ploedereder indicated that there are two distinct situations:

1. One expects to give the USER enough information to recover.
2. One expects to give the TOOL enough information to recover.

The ALS has a list of 15 or so things that can go wrong and would like to find out what's really wrong.

If exceptions are used properly, the error can be narrowed down to one procedure. One of the concerns CAISWG has is the expectation of an implementation to do this. For the most part, CAISWG cannot prescribe the ordering of the checking of exceptions.

The implementors said that there are some interfaces where some things CAN take precedence.

The CAIS Rationale document [4] indicates that NAME_ERROR usually takes precedence.

Again, it was brought out that the current CAIS specification should be reviewed for consistency and clarity. CAISWG recognized the need to cover the second order stuff better, too.

10.0 Alternate Interfaces in the CAIS

The alternate interfaces are in the proposed MIL-STD-CAIS document in order to clarify the semantics of some of the CAIS interfaces. The implementors, for the most part, have also implemented many of the alternate interfaces.

The Alternate Interfaces are the strongest semantic definitions that one can get. The reason why all the exceptions weren't there is that someone would probably come back to nit pick. Initially, CAISWG didn't want the alternate interfaces to be part of an implementation. There are lots of reasons why not to use the alternate interfaces.

The question was then asked, "Why not take out the alternate interfaces?" The fewer interfaces, the better. Questions arise to an implementor if there are several ways to do things.

GOULD thought that the alternate interfaces were just a suggested coding. They use the alternate interfaces a lot, mainly for convenience.

GOULD also thought that some of the string oriented alternate interfaces would probably be more popular. Listing all of the exceptions in the alternate interfaces wasn't needed for the standard. MITRE asked, "Why not do the same thing as was suggested for attributes and relations? Just let the community evolve to the usage of the interfaces." One value of prototypes with tools is to get a frequency count of the usage of these interfaces.

It was pointed out that CAISWG hasn't paid enough attention to the tool writer. The CAIS is not terribly too convenient for the tool writer. The ALS tries to make the KAPSE interfaces as close to the command language as possible. If the toolwriter convenience is necessary, then these alternate interfaces may not be necessary. But in this case, layers of software would be generated. In the ALS, some common interfaces became macros.

Tim Lindquist indicated that the Operational Definition team has implemented the alternate interfaces as they are shown in the specification document. They think that the alternate interfaces will be used, but don't know which ones are the "hot" ones.

11.0 CAIS and the PCTE

11.1 Overview of the PCTE

Herman Fischer presented the overview of the Portable Common Tool Environment (PCTE) and a comparison of the CAIS and the PCTE. This was basically the same presentation given at the KIT/KITIA meeting held in September 1985. These slides are included in these proceedings.

The PCTE Object Management System is based on the Entity-Relationship model like the CAIS. The main similarities of the PCTE to the CAIS include:

1. The node models are nearly identical.
2. The relationship models are nearly identical.
3. The attributes are very similar.

The PCTE does have two function calls for LAN supported distributed processing. The PCTE has true transparent distribution. Nodes and processes, pipes and messages, are fully transparently distributed.

Relationships in the CAIS are unidirectional and n-ary only. However, relationships in the PCTE are bidirectional and the arity is specified by the type.

11.2 Discussion on the PCTE

Discussion on Herman Fischer's overview of the PCTE then followed.

Richard Thall: So, with the PCTE, users can link their own commands into their own CLI. The problem then is that a different search order is used because some commands are linked into the CLI. The command interpreter was originally in the KIT list of items to standardize (it was pointed out that this is not true but that the CAISWG only agreed to discuss it).

The command language is a sensitive interface and shouldn't be addressed lightly. The PCTE, however, is concerned with something that we did not address. This is in respect to the window screen management.

Those interfaces are good for human interaction.

Mike McClimens: What kinds of relationships and attributes are governed by the types (on typed nodes in the PCTE)? What about I/O typing on operations on the

file nodes?

H. F.: We should also pursue process typing.

Richard Thall: I'm skeptical about the typing of operations, of files, and of nodes in the environment. We need a clearcut reason why a typed environment is needed. This is an exceedingly important issue. We need a clearcut requirements definition of a typed environment. This probably should be a future topic of discussion.

H. F.: There is no typing on I/O. Working schemas are inherited by the children. Renaming via move/copy can be done across workstations. There are execution classes for different processors. These are associated with the execution module to map onto the actual processors.

11.3 PCTE

Herman Fischer continued with more information about the PCTE.

Product identity and ownership is confusing as seen by those in the United States. Also, the PCTE can be one of three things. The PCTE is:

1. A set of interface specifications.
2. A prototype extension to the UNIX kernel.
3. A portable piggyback in implemented in the Ada language.

There is no United States participation in the PCTE efforts; especially noticable is the lack of United States AT&T participation.

The UNIX 8th edition is architecturally and functionally compatible and architecturally has implementation differences. The existence of the UNIX 8th edition was a total surprise to the PCTE people. UNIX is a moving target.

The PCTE life cycle is not structured; it is not the way that we in the United States do business. Project management is technically oriented; it is similar to a "corporate research lab" in the United States. There are no marketing considerations. There is an unusually highly trained staff of fourteen people on this project. The PCTE is the second environment worked on by these designers (PAPS was the first one).

Because this is the second time for the development of an environment for most of the

technical staff, the node model IS implementable and a schema on top of it IS implementable!

The PCTE has features that the CAISWG wanted but lacked the resources to work on: distribution and user interfaces. The CAIS should benefit, i.e., grow, from the PCTE experience.

The PCTE Add-on Common Tools (PACT) was then discussed. Its most noticable attribute was that the support for the Ada language was dropped!! It is not known whether support for the Ada language is waning in Europe or whether the commercial people are tied into UNIX.

How do the CAIS and the PCTE relate to each other with respect to the good and bad points? The PCTE is based on a widely used commercially available operating system – UNIX. Their abililty to copy with heterogenous hosts on a network is good. Their node model is good. Herman Fischer thinks that typing is important, but as Richard Thall said, "we must give reasons why it is necessary." The PCTE is good for desk top applications on LANs.

On the interface set, distribution and typing are the main things that they have that the CAIS does not have. They have two primitives for distribution; these two primitives should not be difficult to implement, once included in the CAIS.

11.4 Further Questions on the PCTE

More discussion followed.

LCDR Philip Myers: My impression is that the way they're doing this is rather loose; there is no worry about ownership, documentation, etc. Is this the way they do business over there?

H. F.: Documentation is indeed performed on this project, but not as formal as DoD. Documentation is done in a rather informal way, but it IS done. However, don't think that informal documentation impacts the quality of the product.

Richard Thall: The existance and use of the Ada language changes the way you deal with B5s and C5s. Pseudo-code to code is different than the normal paradigm. The government realized that standards are getting outmoded and are improving those standards.

H. F.: The Emeraud people had informal pseudo-code that stays back as comments.

Erhard Ploedereder:
What is Portable about the PCTE?

H. F.: There is a binary license on UNIX for the kernel, etc. It is really compatibility rather than portability [illegible] is the requirement (compatibility so that existing tools can be ported into the environment). Machines that the PCTE is to implemented include: Bull SPS-7, 3B2, PCS, Perq 3, Sun, and Apollo. There is not a great deal of incentive to bring it up on the VAX, but on several workstations. It is portable to a machine that runs the System V kernel (with certain assumptions).

Erhard Ploedereder:
Then, it's a nice extension to System V and that's basically all.

Jack Kramer: The argument is that you could do the same thing to any operating system, e.g., VMS.

There are two separate issues.

1. We need support tools for COBOL and other languages.

2. If we don't capture what's out there, we'll lose a lot which is really out there.

H. F.: Applications in COBOL must be preserved.

Jack Kramer: This is an example that many companies (IDA included) are struggling with right now. The same machine shares all kinds of applications – software development, accounting, etc. We want to preserve all this software, including financial applications, etc. As an evolutionary step, it's something that you want to deal with.

Richard Thall: The CAIS itself is oriented to software development. If we're going to extend it to other areas (WIS, Space Station, etc.), we must examine our basic assumption. Maybe we need typed stuff in these cases, then.

Jack Kramer: There is a difference between content management and other configuration management. In a sense, we are mapping from an object to particular processes, e.g., DBMS. There is a problem of applying old tools to objects created by new tools.

H. F.: Even the PCTE has that problem.

Richard Thall: Any environment that does not validate the stuff coming into that environment will crash. The nature of the Ada program is different from COBOL programs in its dynamism. The real problem of building FORTRAN or COBOL interfaces is with the exceptions turned into error codes, etc. In the evolution of the ALS, it returned error codes because the interfaces were really PASCAL.

12.0 CAIS Usability and Implementability

Discussions then moved into the area of the implementability of the CAIS and about the useability of the CAIS from the end users.

LCDR Philip Myers began the discussion with: If we ask the implementors to do the hard things, we're looking for making life easier for the developers and for the maintainers downstream.

Asked if the implementors have looked into moving their development efforts onto the CAIS, MITRE and GOULD replied that their respective implementations are not yet mature enough for that to be done at this time. Also, there is no Ada compiler available (with interfaces to the CAIS).

12.1 CAIS Documents

A general discussion followed concerning the CAIS document itself. Many expressed the sentiment that examples are needed in the document. An Implementors Guide is probably useful – there is a need for it. The Reader's Guide [9] will be available in the short term (the DRAFT Readers Guide is currently being reviewed). The CAIS Rationale document [4] will answer many of the why and why not questions, but won't be available for several months (a rough draft of the CAIS Rationale has been generated since the Workshop). The CAIS specification document shouldn't be a user's manual.

It is not clear how predefined attributes are handled in the specification document. They are spread out too much in the document. Which operations can be applied to the predefined attributes? Do the defaults replace or add to what's there already? A table of predefineed attributes and operations that affect them is needed.

These are the issues which some sort of CAIS documents must address.

12.2 CAIS Conformance Testing

Tim Lindquist also indicated that what is needed is an explicit statement concerning access control at some of the procedure interfaces. Discretionary access checking at the OPEN procedure may be an efficiency concern.

LCDR Philip Myers remarked that creation of things in the security world takes forever,

anyway.

The Ada Language compiler's ACVC process is a good model. Operational tests are "proof of the pudding". Implementors can debate any ACVC test. Then it's debated among a Fast Reaction Team to determine whether the test or the implementation is right.

Ray Szysmanski, chair of the Evaluation and Validation Team, brought the Workshop attendees up-to-date on various efforts funded by the E&V Team. On October 11, 1985, an RFP went out for the CAIS Validation Capability (CVC). Funding exists for this project through 1989. However, he couldn't say when test versions will be offered.

As implementors watched over the evolution of the ACVC, in a similar manner, implementors can put any test into the test suite for the CAIS Validation Capability.

The possibility of a quarterly progress report at E&V Team meetings is good. They hope to be under contract in March or April 1986. This will be for an operational test set for the CAIS.

The implementors need to get answers back from the CAIS designers quickly. They also would like to have Binding Interpretations similar to those of the Ada Language Maintenance Committee.

Currently, the SIGAda CAIS Implementors Working Group uses the account CIG-INFO@MITRE to communicate among members of that community. Anyone who would like to be added to this mailing list should send a request to CIG-RQST@MITRE or to HOWELL@MITRE or to CROBY@ADA20.

Tim Lindquist indicated that the constant limits need to be set properly. The behaviour should be specified outside the limits so that the CVC could examine the behaviour. These constants are lower bounds.

GOULD's main areas of concern include: Access Control, LIST_UTILITIES, and Input/Output. However, they have been in close contact with the CAISWG people during their implementation.

Jack Kramer remarked that with the lessons learned from this Workshop, the implementors should get together more often. A list of the first 20 or so issues shoud be generated so that

when the implementors get together again, they can discuss these specific issues more thoroughly.

12.3 Access Control

It is difficult to read and understand access control the way that it is presented in the CAIS document.

GOULD is not implementing Mandatory Access Control. They are implementing Discretionary Access Control exactly as in the CAIS document.

Jack Kramer, chair of the CAISWG, encouraged everyone to look into the implementation strategies for discretionary access control. These may include cacheing strategies.

GOULD also mentioned that, in LIST_UTILITIES, there is a lost facility (from earlier versions of the CAIS) to mix named and positional parameters. In their CLI, they wanted to follow the ALS idea of an Ada shell, but had to fudge around it in order to simulate it.

They open about six files at node OPEN time. The main reason comes down to discretionary access control. They can cache the whole lot at startup time. They've also noticed that the CAIS specification has constantly been evolving in the area of access control.

Once a complete implementation is done, everyone can step back and examine the efficiency issues with respect to access control.

LCDR Philip Myers said that, in practice, these (security-related items) are necessary overheads. In a real-world environment, there's a big overhead.

GOULD cautioned that user nodes should not be manipulated inside the CAIS implementation at all.

MITRE said that they have been using (an implementation of) the system node for adding users in real time. Adding users is outside the CAIS but it still affects user nodes in running the CAIS.

The USER relation was supposed to be a special situation. However, it is not that terrible a burden for MITRE's implementation.

The system node was added only for a formal axiomatic definition of the CAIS. It is there mainly for consistency and for logic, but that's its limit.

Richard Thall asked Gould: When processes are initiated, are the process nodes kept in the database? They responded: Yes, all nodes are on disk to obey the CAIS specifications literally, but later we will make it a daemon-like thing [in the UNIX sense].

13.0 Tools in the CAIS

One of the topics brought out by the CAIS designers was tools and their availability.

Jack Kramer: Are there any problems with tools?

Pete Carr: Tools are fine but that's probably due to the Ada language.

Robert Stevenson: Most tools are I/O oriented. We don't have complex tools that really maniuplate the node model. A minor problem in the GOULD environment is the sheer size. Is the CAIS expected to fit on a workstation?

Herman Fischer: I expect to see a CAIS implementation on PC workstations with 5 megabtypes of memory and a Winchester disk.

LCDR Philip Myers: I hope that we fend off the "all things to all people" syndrome.

Mike Doub: Concerning the importing and the import/export of tools into the CAIS, the CAIS specification says to copy the data into it. We are concerned about editing a file within the CAIS, then getting that file back outside of the CAIS in order to be compiled by an outside Ada compiler, and then bringing it back into the CAIS.

Jack Kramer: CAISWG felt that import/export interfaces had to be in there. I'm surprised that the word "copy" is explicitly stated there.

Tim Harrison: The intent was to make the file available so that the CAIS is consistent.

Mike Doub: We read "copy" as "copy byte by byte".

Jack Kramer: Consistency is an issue.

Mike Doub: It needs to be clarified semantically; it should be reworded at a minimum.

LCDR Philip Myers: The intent is to have an object made unknown to the CAIS while the object is manipulated outside of the CAIS and then vice versa.

Tim Harrison: When something comes into the CAIS world, this implies the addition of CAIS attributes, etc.

Jack Kramer: As long as the CAIS node structure is maintained consistently, a physical copy is not really implied.

Richard Thall: There should be a fairly large chapter in the Implementors Guide about the relation of the CAIS world and the non-CAIS world.

Jack Kramer: In a pure world, my view of an implementation of the CAIS in the near term will have to address differences in tools and data between the CAIS database and the non-CAIS database. Where you have to mix two worlds, the system manager will have to impose some rules. Implementations will have to give some help during this time frame. During the transition period, we'll have to live with system management. The CAIS node structure must be consistent.

Mike Doub: MITRE thinks that something should be changed in the standard when running other tools brought in under the CAIS. We're going to have an agent to use the tools without physically importing the tool. Effectively, this gives foreign tools their own little world.

Jack Kramer: Tools that have a database model in their architecture will not work.

Richard Thall: The kicker is the INCLUDE statement in a tool (compiler, etc.). Once a file name is in the source, then you have a problem.

Mike Doub: This is an unsolvable problem in general.

Richard Thall: You can't take that agent thing too far.

Mike Doub: The other extreme is to trap operating system calls. That's no good. The real solution is to have adequate tools out post-haste.

Richard Thall: For the government, the government should always have the source available. I see a problem in that people want to use existing DBMSs, graphics packages, etc.

The problem is even worse in secure environments. We can't tolerate that sort of thing. A lot of these fancy things have a difficult time living in the Ada RTS – most probably you will have some hooks into the operating system. This is a very technical sticky issue. Ada's RTS makes heavy use of VMS in the ALS.

Jack Kramer: The Ada model was designed with not too much look at an implementation on existing operating systems.

Richard Thall: Mike Kamrad's ARTEWG is looking at these issues.

Jack Kramer: Rich, if you had your choice of ten ALS tools to exercise the CAIS, what would they be?

Richard Thall: The list would include the Command Language Interpreter, any tool dealing with Ada program libraries, the compiler, the linker, the debugger, and an exporter. The ALS data structure is great until you get to program libraries.

Mike Doub: What you're talking about is the development of an Ada compilation system

on a CAIS implementation.

Jack Kramer: The Air Force's AIE (Ada Integrated Environment) is the closest environment to the CAIS of all the environments produced thus far.

Richard Thall: I would like to see a configuration management system built on top of the CAIS to support baselines, etc. It would be a really good exercise to let a contract to do a configuration management system on top of the CAIS. I think that it is very possible.

The real issue is controlling baselines, mixing users, etc.

Jack Kramer: It would definitely be a good exercise to do a configuration management system on top of the CAIS. Conceivably it could be done today.

14.0 Performance in the CAIS

What are acceptable levels of performance in the initial implementations of the CAIS? People will accept some startup overhead. If a tool operates poorly, users won't accept it.

14.1 Historical performance of the ALS

Richard Thall discussed some of the intial performance-related issues of the Army's Ada Language System (ALS).

In the editors, once the I/O for opening a file is completed, the actual I/O goes okay. Access to attributes is slow but not on a single attribute basis. The way tools are used, the access attributes are fetched, everything on the path is then gotten, etc. – this adds up to 30-40 disk accesses per level in the ALS. There are two things to worry about with respect to performance: how slow the OPEN is, and how many operations have to be done. In the CAIS, the latter is going to kill you.

About 60-70% of any ALS degradation is done by the Ada language support tools (mostly in compilation and linking) and the other 30% or so is done by examination of attributes, etc. But there is a better way of handling the Ada language, e.g., Rational's machine.

Everyone perceived that the cost of using the Ada language would come down in the future.

It's that programming-in-the-large is now coming into current usage. There will be more recompilations in the Ada language support systems than in those systems that support most other languages. That is a penalty of Ada language usage. Compilation speeds will get a little better, though.

14.2 Perceived Response Time

PC's will permeate and day-to-day keystrokes will be done on PC's. Configuration management will be taking care of uploading/downloading.

There is a perceived response time. Some tools in the ALS have been modified to spit out intermediate status results. The edit-compile-bind-execution-edit circle model has the most time spent in the compile-bind portion; thus the whole circle is slow.

LCDR Philip Myers (thinking in maintenance mode) indicated that programmers will be working on several programs. They can upload/download different programs' portions. Yes, there will be a management problem concerning the discipline of programmers. Programmers want to be productive.

If users don't see any response after initiation within about 6 seconds or so, they will become discouraged. If something takes about 1/2 hour, then they can switch universes. If it only takes about 5-10 minutes, they won't.

There's a time span that a human will know about regarding productivity. If a wait is above this number, the person will put the job off to the background. If the wait is perceived to be below this number, the person will wait for it.

14.3 Prototype CAIS Implementations

Robert Stevenson, manager of the CAIS project at GOULD, said that the perception at GOULD is that if the CAIS is out and it doesn't meet certain performance requirements, it will hurt the CAIS effort more than help it.

Jack Kramer said that early implementations should be sold to management as a prototype. It is a serious concern. But make sure that people understand what it is. Rapid prototyping efforts are being done here -- just make sure that it is understood that it is indeed a rapid prototyping effort.

We are now moving into programming-in-the-large today. Time delays are a fact of life in those environments.

Richard Thall, when asked if there would be optimized environments, responded that the Rational Ada environment is an example of one, and best perhaps as an extreme example.

GOULD indicated that if the CAIS was a standard five years ago, they would have put it on a bare machine. GOULD also indicated that their challenge is to get a tool to run under the CAIS only slightly slower than the tool runs under its native environment.

There is an expectation that dedicated hardware that does nothing but support the Ada language will be available in the future.

Jack Kramer said that, with the CAIS node model, most existing file systems are subsets of the CAIS. If a standard like the CAIS exists, then there will be an evolution of operating systems going in that direction. A lot of companies will want to sell hardware in order to get more efficient implementations.

14.4 Perceived Performance

Richard Thall came back to the problem of perceived performance: why are workstations attractive? If it's the user's own, it is his, no one else's. We'll see stuff on workstations in the future. So we'll see the CAIS on a workstation, some people say. I don't think so but some say that there will be CAIS servers on LAN's. We need to work on the user interface; CAIS interfaces will be on local PC's.

The combination of the CAIS on a local PC doing local things and periodically going out to the big machine to get something will increase productivity and perceived performance.

CLI functions should be on the local machine (no inference made that the CAIS be distributable). Concerning issues of perceived performance, in order to decide what a local CAIS exports, it is recognized that certain high speed tasks are done locally. Support is needed in the CAIS once we recognize that user interfaces are of special concern.

15.0 User Interfaces and General CAIS Input/Output

If CLI functions should be on the local machine, then support is needed in the CAIS once it is recognized that user interfaces are of special concern. Interfaces for window/graphical management should also be defined.

CAIS 2.0 addresses these problems.

An issue before the community now is what is today's CAIS as a stepping stone. There has been some reluctance about having I/O in the CAIS, but everyone said that Ada I/O was needed because 90% of the tools use I/O 90% of the time.

I/O is a major tool portability issue. That's why it's in the CAIS today. The current I/O should be fixed. Additional packages can be added once it is understood what the community would generally support. What is in the CAIS today is what today's tools can support.

We're experiencing a more rapid evolution than originally expected.

Tool transportability should be tested. In the early part of next year, that can be done with (at least) the implementations represented here.

16.0 Recommendations

Many recommendations came out of this Workshop.

1. A different means of referring to CAIS 2.0 would be appropriate at this time.

2. The CAIS does not have a Mandatory Secure Model and there was a perception that any Mandatory Secure Model can be inserted in the CAIS. The question "Can the CAIS be implemented on a bare system and be secure?" must be addressed.

3. Clearer semantics must be included in the CAIS document. The implementors would like to have interactions or the lack thereof among different interfaces explicitly stated.

4. There should be something concerning naming in the CAIS about distribution.

5. The requirement for security as stated in the Requirements and Design Criteria document for the CAIS [8] must be firmed up.

6. The type definitions of the CAIS should be moved to the CAIS package level, i.e., to the outermost package level.

7. All limited private types should be accessible in any of the CAIS implementations.

8. It would be a good exercise to define the CAIS node attributes and relationships using an external representation.

9. An import/export of the node structure itself would also be a useful exercise.

10. Issues of archiving must be addressed and solutions enumerated.

11. Examples are needed in the CAIS document; these should be generated in the next version.

12. A table of predefined attributes and operations (procedures) that affect them should be placed in the CAIS document.

13. An explicit statement concerning access control at some of the procedure interfaces should be included in the CAIS document.

14. The CAIS implementors need to get answers back from the CAIS designers very rapidly. Some sort of mechanism must be created in order to satisfy this need.

Appendix A
LIST OF PARTICIPANTS

CAIS Working Group (CAISWG) of KIT/KITIA

Herman Fischer
Litton Data Systems
MARK V
16400 Ventura Blvd
Suite 303
Encino, CA 91436
(818) 995-7671
HFISCHER@Ada20

Tim Harrison
Texas Instruments, Inc.
P. O. Box 801
Mail Stop 8007
2501 W. University
McKinney, TX 75069
(214) 952-2152
HARRISON%TI-EG.CSNET@CSNET-RELAY

Tim Lindquist
Computer Science Department
Arizona State University
Tempe, AZ 85287
(602) 965-2783
LINDQUIS%ASU.CSNET@CSNET-RELAY

Hans Mumm
Naval Ocean Systems Command
Code 423
San Diego, CA 92152
(619) 225-6682
MUMM@NOSC-TECR or HMUMM@Ada20

LCDR Philip Myers
Ada Joint Program Office
The Pentagon
3D139 (1211 Fern Street)
Washington, DC 20301-3081
(202) 694-0209
MYERS@NRL-CSR

Erhard Ploedereder
Tartan Laboratories
477 Melwood Avenue
Pittsburgh, PA 15213
(412) 621-2210

PLOEDEREDER@TARTAN

Carl Schmiedekamp
Naval Air Development Center
Code 5033
Warminster, PA 18974
(215) 441-1779
SCHMIEDE@NADC

Ray Syzmanski
AFWAL/AAAF
Wright Patterson AFB
Dayton, OH 45433
(513) 255-6730
RSZYMANSKI@Ada20

Richard Thall
SofTech, Inc.
460 Totten Pond Road
Waltham, MA 02254-9197
(617) 890-6900 ext 313
SOFTKIT@Ada20

CAIS Implementors Group

Gould, Inc.
C. S. D.
6901 West Sunrise Blvd.
Fort Lauderdale, FL 33321
RSTEVENSON@Ada20

Pete Carr
(305) 587-2900

Mike Doub (Manager, Ada Project Development)
(305) 587-2900 ext 3891

Robert Stevenson (Manager, CAIS Project)
(305) 587-2900 ext 3823

IBM
Federal Systems Division
9500 Godwin Drive
102/075
Manassas, VA 22110

Jeff Vermette

(703) 367-6715
DRAKE@Ada20

MITRE Corporation
1820 Dolly Madison Blvd.
McLean, VA 22102

Helen Gill
(703) 883-7980
GILL@MITRE

Chuck Howell
(703) 883-6080
HOWELL@MITRE

Robbie Hutchison
(703) 883-7037
HUTCHISO@MITRE

Mike McClimens
(703) 883-7697
MIKEMC@MITRE

Tana Reagan
(703) 883-6547
REAGAN@MITRE

Software Engineering Institute
Carnegie-Mellon University
Pittsburgh, PA 15213

Dan Burton
DBURTON@CMU-SEI
(412) 578-7613

Robert Ellison
ELLISON@CMU-SEI
(412) 578-7705

Peter Feiler
FEILER@CMU-SEI
(412) 578-7700

Dan Miller
DHM@CMU-SEI
(412) 578-7616

William G. Wood
WGW@CMU-SEI
(412) 578-7723

WIS Operating System Task Force

Mike Liu
Department of Computer and Information Science
Ohio Statue University
2036 Neil Avenue Mall
Columbus, OH 43210
(614) 422-1837
LIU%OHIO-STATE.CSNET@CSNET-RELAY

John (Jack) Stankovic
Dept. of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213
(412) 578-7678
STANKOVIC@CMU-SEI

Institute for Defense Analyses
Computer & Software Engineering Division
5 Skyline Place
5111 Leesburg Pike, Suite 300
Falls Church, VA 22041

John Chludzinski
(703) 824-5518
JCHLUDZINSKI@Ada20

Jeff Clouse
(703) 824-5514
JKRAMER@Ada20

Jack Kramer
(703) 824-5504
KRAMER@Ada20

Clyde Roby
(703) 824-5536
CROBY@Ada20

Appendix B
GOALS

GOALS
of the
CAISWG/CIG/SEI
Meeting

1. Exchange of information between the designers of the CAIS (members of the CAIS Working Group (CAISWG) of the KIT/KITIA) and those groups (mainly represented by the CAIS Implementors Groups (CIG) of SIGAda) actually doing prototype or development of the CAIS resulting in a set of issues that need to be addressed and possibly acted upon during the evolution of the MIL-STD-CAIS.

2. Introduction of other environment ideas (most notably UNIX and PCTE) to the CAISWG and CIG for possible inclusion into the CAIS during its evolution.

3. Discussion of the issues.

Appendix C
AGENDA

Tuesday, October 29th

9:00 AM Welcome
Registration ($20)

General Introduction of Attendees

Statement of Goals

Generation of Working Agenda

DISCUSSION OF ISSUES

- Implementation Issues
- CAIS and PCTE
- CAIS Package Structure
- CAIS and Security Issues
- WIS Distributed OS
- Deferred Topics for Space Station Needs

Wednesday, October 30th

9:00 AM Administrivia

Continuation of DISCUSSION OF ISSUES

Lunch Breaks will occur about 12:00 noon. Dinner Breaks will occur about 5:00 PM. Mid-morning and Mid-afternoon breaks will occur approximately 10:30 AM and 3:30 PM respectively.

Evening sessions MAY occur as needed.

Appendix D
ISSUES

This was the list of issues that was generated before the Workshop met; it was distributed at the Workshop.

D.1 General Issues

1. A thorough discussion is needed of how implementors are going to deal with the CAIS security requirements on hosts that do not implement multilevel security as part of the native host architecture.

2. Discussions on the current work elsewhere that may be related to the CAIS. This may include enhancements to UNIX that either relate to CAIS features or suggest additional services the CAIS might be extended to provide; Microcomputer Operating System Interfaces (MOSI); how the CAISWG sees the CAIS impacting The STARS SEE, the Space Station SEE, SEI showcase environments, etc. Also included here may be issues concerning how the PCTE could be used as an implementation vehicle for the CAIS and what features the PCTE has that the CAIS does not that perhaps should be added to the CAIS.

3. Discussion of the overall design model behind LIST_UTILITIES. There must be a better solution to the problem of returning strings from LIST_UTILITIES routines.

4. Discussion of the package structure of the CAIS specifications, including changes to the I/O packages and suggestions for a new global exception package.

5. Additional iterator facilities should be added.

6. The semantics are not fully defined in many areas. One of the major areas of concern is in situations where more than one exception could be raised and no guidance is given on choosing a "proper" one.

7. Clarifications should be made where it is not clear that the issues are well understood, e.g., related to the semantics for dependent processes, and some interfaces that appear to be redundant in nature.

D.2 WIS Distributed OS Issues

There was some concern about problems with secondary relationships.

There does not appear to be requirements on the consistency of secondary relationships, so ensuring consistency will not be a problem, although it might be for users of CAIS. Has there been any thought given to consistency among secondary and primary relationships? To give a database example of this, primary relationships might put a set of nodes in a given

department (or directory) with secondary relationship "REPORT_TO" pointing to the manager with the manager having a "MANAGES" relationship that is inconsistent with "REPORTS_TO". Perhaps this would be more convincing with a technical department with separate managers for administrative and technical matters. This is an issue with their use, not their implementation.

In implementing secondary relationships, the assumption was made that a secondary relationship can be stored at the "base node" in the CAIS terminology. Then one issue that was seen is whether a secondary relationship can reasonably be forgotten when the base node is deleted. The assumption was made that this is reasonable yet the paragraph just before 5.1.2.22 in the CAIS document says that secondary relationships must remain until they are explicitly deleted; that is only secondary relationships "to" such a node, as opposed to "from" a deleted node. Thus, in the implementation envisioned for a distributed system, the storage of secondary relationships is distributed, with each secondary relationship stored at the storage location for the base node. The only exception to this that can be seen at this point is a possible decentralized relationship mechanism using multicast. As the naming design is developed, it is hoped that there may be some clarification on this point.

D.3 CAIS and the Space Station

The following is quoted from a paper that was presented at the AIAA conference in Long Beach in October 1985 [10].

> The complete resolution of at least three areas which are vital to the context of the Space Station has been deferred until later revisions of the CAIS. Quoting from the current draft,
>
> ''c. Distributed environments ... currently deferred is the decision whether or not to provide to the user explicit CAIS interfaces to control the distribution of the environment, including designation of where nodes exist and where execution takes place. Note that a set of distributed processors could include one or more target machines.''
>
> ''d. Inter-tool interfaces. The current CAIS does not define ..., the command processor language syntax, ... or the interaction between the run time system and debugger tools. Currently deferred are decisions regarding ... and whether or not to place constraints on the run time system to provide

process execution information.''

''e. Interoperability. The current CAIS ... does not define external representations of data for transfer between environments or between a host and a target.''

In the process of evolving later versions to meet the needs of distributed applications, it will be necessary to decompose the process node of the current version to support a finer grain of program element representation. The process node in conjunction with a queue node provides a meta level representation of all program units of an Ada program. Thus if the program to be executed will spawn three tasks, each to be executed simultaneously on a different processor which is remote from the processor responsible for the parent rpocess, the currently proposed process node (with its associated queue node) would be the single required CAIS repository of information on the state of the program, resources being utilized, etc. It seems obvious that for purposes of debugging, performance analysis, dynamic reconfiguration support and many other features that characterize distributed applications, a finer grain of representation is needed.

Appendix E
MITRE's LIST_UTILITIES Package

Below is a revised specification for the CAIS Package LIST_UTILITIES. Areas which have been modified are identified in the listing.

Mike McClimen's comments on the LIST_UTILITIES package are:

"My impression has been that the other implementors have found the LIST_UTILITIES interface as cumbersome as we have. In Pittsburgh it seemed that the CAISWG shared that feeling (or at least its possibility) and that the expanded token approach was favorably received. The complete update of Section 5.4 will take considerable work. I am willing to work towards that update but would like to coordinate closely with those making the decision on any changes to the CAIS."

"In Pittsburgh, two statements were made which were not part of my understanding when I wrote the new specification. They result in a better interface and have only minor impact on the revised specification. They are:"

* the external representation of string_items require quotes only when they appear as part of a list. As individual items they are unquoted.

* float and integer items are intended to be stored internally so that they may be retrieved under any generic representation (float or integer, respectively) and constraint errors are raised only when the value of the item violates constraints of the type.

"I would like to hear people's opinions on the proposed subprogram names and on the proposal to eliminate the string interface for Named parameters, requiring a TO_TOKEN function call."

MITRE's proposed LIST_UTILITIES follows.

```
------------------------------------------------------------------
--  This package specification is a strawman proposal for the
--  redefinition of List Utilities based upon an expanded
--  Token_Type.  The new Token_Type is able to represent
--  integers, floats, identifiers, strings, and lists.  In this
--  strawman, changes required to expand Token_Type are indicated
--  by comments.
--
--  In summary, the proposed interface changes are:
--
--      1.  Sections 5.4.1.1  thru 5.4.1.7
--                      unchanged list operations
--      2.  Sections 5.4.1.8   and 5.4.1.13
--                      delete and use token operations
--      3.  Sections 5.4.1.9  thru 5.4.1.15
--                      unchanged list operations
--      4.  Sections 5.4.1.16 thru 5.4.1.19
--                      change to operate on tokens instead
--                      of list_items. Add Token function
--                      to specify any list_item as a Token
--      5.  Sections 5.4.1.20
--                      supply To_Token and Value for all item
--                              kinds
--                      supply Token function
--                      supply Text_Length and Copy for
--                              Token_Types
--                      delete the Insert, Replace, Extract, and
--                              Position_By_Value subprograms
--                              since these are now supplied in
--                              16-19 for all item kinds
--      6.  Sections 5.4.1.21 thru 5.4.1.23
--                      delete and use token operations
--
--  Changes to our data structures to implement an expanded
--  token are shown for reference only.
--
--  Additional changes(recommended):
--
--      The interface would be simplified if the subprograms with
--      the parameter Named of type Namestring were eliminated.
--      This would require:
--              Extract(List,Token("The_Name"),List_Token);
--      instead of:
--              Extract(List,"The_Name",List_Token);
--
--      Function names have been shortened as follows:
--              Token   for     To_Token
--              Value   for     To_Value
--              Text    for     To_Text
--              Element for     new function to treat a list
--                              element as a token
```

```
--------------------------------------------------------------------

package List_Utilities is

   -- The following type and exception declarations are from
   -- CAIS 5.4.1.1

   type List_Type  is limited private;
                                                        --**
   type Token_Type is limited private;                  --**EXPANDED
                                                        --**
   subtype Namestring is string;

   type List_Kind is (Unnamed, Named, Empty);
                                -- See note above re "empty"

   type Item_Kind is (List_Item, String_Item, Integer_Item,
                      Float_Item, Identifier_Item);

   subtype List_Text      is String;
   type    Count          is range 0 .. Integer'Last;
   subtype Position_Count is Count range
                                Count'first+1 .. Count'last;

                                        --**
   Search_Error   : exception;          --**SHOULD BE DELETED
                                        --**

   EMPTY_LIST     : constant List_Type;


   -- MIL_STD CAIS 5.4.1.2
   procedure Copy (To_List      : out List_Type;
                   From_List    : in  List_Type);

   -- MIL_STD CAIS 5.4.1.3
   procedure To_List (List_Literal : in List_Text;
                      List         : out List_Type);

   -- MIL_STD CAIS 5.4.1.4
   function Text (List : in List_Type) return List_Text;

   -- MIL_STD CAIS 5.4.1.5
   function Is_Equal(List1 : in List_Type;
                     List2 : in List_Type)
                     return  Boolean;


   -- MIL_STD CAIS 5.4.1.6
   procedure Delete (List     : in out List_Type;
```

```
                    Position : in Position_Count);

procedure Delete (List     : in out List_Type;
                  Named    : in Namestring);

procedure Delete (List     : in out List_Type;
                  Named    : in Token_Type);


-- MIL_STD CAIS 5.4.1.7
function Get_List_Kind (List : in List_Type)
                        return List_Kind;


                                          --**
-- MIL_STD CAIS 5.4.1.8                   --**COULD BE DELETED
                                          --**
function Get_Item_Kind (List     : in List_Type;
                Position : in Position_Count)
                       return Item_Kind;

function Get_Item_Kind (List     : in List_Type;
                Named    : in Namestring) return Item_Kind;

function Get_Item_Kind (List     : in List_Type;
                Named    : in Token_Type) return Item_Kind;

-- MIL_STD CAIS 5.4.1.9
procedure Splice(List       :in out List_Type;
                 Position   :in     Position_Count;
                 Sub_List   :in     List_Text);

procedure Splice(List       :in out List_Type;
                 Position   :in     Position_Count;
                 Sub_List   :in     List_Type);


-- MIL_STD CAIS 5.4.1.10
procedure Merge (Front    : in List_Type;
                 Back     : in List_Type;
                 Result   : in out List_Type);

-- MIL_STD CAIS 5.4.1.11
function Set_Extract(List      : in List_type;
                     Position  : in Position_Count;
                     Length    : in Positive := Positive'Last)
                     return        List_Text;

-- MIL_STD CAIS 5.4.1.12
function Length (List : in List_Type) return Count;
```

```
    -- MIL_STD CAIS 5.4.1.13
    function Text_Length (List : in List_Type)
                          return Positive;--Mod to MIL_STD

                                          --**
                                          --**NO LONGER NECESSARY
                                          --**
    function Text_Length (List     : in List_Type;
                          Position : in Position_Count)
                          return       Positive;

    function Text_Length (List     : in List_Type;
                          Named    : in Namestring)
                          return       Positive;

    function Text_Length (List     : in List_Type;
                          Named    : in Token_Type)
                          return       Positive;

    -- MIL_STD CAIS 5.4.1.14
    procedure Item_Name (List       : in List_Type;
                         Position   : in Position_Count;
                         Named      : out Token_Type);

    -- MIL_STD CAIS 5.4.1.15
    Function Position_By_Name(List  : in List_Type;
                              Named : in Namestring)
                              return    Position_Count;

    Function Position_By_Name(List  : in List_Type;
                              Named : in Token_Type)
                              return    Position_Count;

    -- MIL_STD CAIS 5.4.1.16
    procedure Extract (List       : in List_Type;
                       Position   : in Position_Count;
                       List_Item  : out Token_Type);
                                              --**Token_Type

    procedure Extract (List       : in List_Type;
                       Named      : in Namestring;
                       List_Item  : out Token_Type);
                                              --**Token_Type

    procedure Extract (List       : in List_Type;
                       Named      : in Token_Type;
                       List_Item  : out Token_Type);
                                              --**Token_Type

            --**
            --** Element function added to identify token in
```

```
        --** a list.  This allows removal of list operations
   function Element (List       : in List_Type;
                     Position   : in Position_Count)
                     return Token_Type;       --**Token_Type

   function Element (List       : in List_Type;
                     Named      : in Namestring)
                     return Token_Type;       --**Token_Type

   function Element (List       : in List_Type;
                     Named      : in Token_Type)
                     return Token_Type;       --**Token_Type

   -- MIL_STD CAIS 5.4.1.17
   procedure Replace (List_Item  : in out List_Type;
                      List       : in     Token_Type;
                                                --**Token_Type
                      Position   : in Position_Count);

   procedure Replace (List_Item  : in out List_Type;
                      List       : in     Token_Type;
                                                --**Token_Type
                      Named      : in Namestring);

   procedure Replace (List_Item  : in out List_Type;
                      List       : in     Token_Type;
                                                --**Token_Type
                      Named      : in Token_Type);

   -- MIL_STD CAIS 5.4.1.18
   procedure Insert  (List       : in out List_Type;
                      List_Item  : in Token_Type;
                                                --**Token_Type
                      Position   : in Count);

   procedure Insert  (List       : in out List_Type;
                      List_Item  : in Token_Type;
                                                --**Token_Type
                      Named      : in Namestring;
                      Position   : in Count);

   procedure Insert  (List       : in out List_Type;
                      List_Item  : in Token_Type;
                                                --**Token_Type
                      Named      : in Token_Type;
                      Position   : in Count);

  --MIL_STD CAIS 5.4.1.19
  function Position_By_Value
                (List           : in List_Type;
                Value           : in Token_Type;
```

```
                                        --**Token_Type
            Start_Position  : in Position_Count
                                        := Position_Count'First;
            End_Position    : in Position_Count
                                        := Position_Count'last)
            return      Position_Count;


--MIL_STD CAIS 5.4.1.20
package Identifier_Items is
        -- MIL STD CAIS 5.4.1.23.1
                                                --**Token_Type
        function Token(Identifier : in Namestring;
                       String     : in boolean)
                       return Token_Type;
                                                --**Token_Type
        function Token(List       : in List_Type)
                      return Token_Type;
                                                --**Token_Type
        generic
              type Integer_Number is range  <>;
        function Integer_Token(List_Item  : in Integer_Number)
                              return Token_Type;
                                                --**Token_Type
        generic
              type Float_Number   is digits <>;
        function Float_Token(List_Item  : in Float_Number)
                             return Token_Type;




                                                --**Token_Type
        procedure To_Token(Identifier : in Namestring;
                           Token      : in out Token_Type;
                           Is_String  : in boolean);
                                                --**Token_Type
        procedure To_Token(List       : in List_Type;
                           Token      : in out Token_Type);
                                                --**Token_Type
        generic
              type Integer_Number is range  <>;
        procedure Integer_To_Token
                      (List_Item  : in Integer_Number;
                       Token      : in out Token_Type);
                                                --**Token_Type
        generic
              type Float_Number   is digits <>;
        procedure Float_To_Token
                      (List_Item  : in Float_Number;
                       Token      : in out Token_Type);
```

```
        -- MIL_STD CAIS 5.4.1.20.2
        function Text(List_Item : in Token_Type)
                         return         Namestring;
                                              --**
                                              --**ADDED
                                              --**
        function Text_Length(List_Item : in Token_Type)
                              return         Positive;


        -- MIL_STD CAIS 5.4.1.20.3
        function Is_Equal(Token1 : in Token_Type;
                          Token2 : in Token_Type)
                          return      boolean;
                                              --**
                                              --**ADDED
                                              --**
        procedure Copy(Token1 : in out Token_Type;
                       Token2 : in     Token_Type);

        -- MIL STD CAIS 5.4.1.20.4
        function Value(List_Item : in Token_Type)
                                    --**Token_Type for
                                    --strings, identifiers
                         return         Namestring;

        generic
              type Integer_Number is range  <>;
        function Integer_Value(List_Item : in Token_Type)
                               return      Integer_Number;
                                              --integers

        generic
              type Float_Number is range  <>;
        function Float_Value(List_Item : in Token_Type)
                             return        Float_Number;
                                              --floats

        function Value(List_Item : in Token_Type)
                         return         List_Type;
                                              --lists


        --MIL_STD CAIS 5.4.1.20.4 through 7  deleted

end Identifier_Items;

--MIL_STD CAIS 5.4.1.21                      deleted
--MIL_STD CAIS 5.4.1.22                      deleted
--MIL_STD CAIS 5.4.1.23                      deleted
```

```
private

    type Item_Descriptor;
    type List_Type is access Item_Descriptor;

    type String_Of_Any_Length(
        Size        : positive) is
    record
        Kind        : Item_Kind;
                          --***moved from Item_Descriptor***
        List        : List_Type;
                          --***moved from Item_Descriptor***
        Value       : string(1..Size);
    end record;
    type Token_Type is access String_Of_Any_Length;

    type Item_Descriptor is
    record
      --Kind        : Item_Kind;    ***moved to Token_Type***
        Name        : Token_Type;
        Element     : Token_Type;
      --List        : List_Type;    ***moved to Token_Type***
        Next_Item   : List_Type;
    end record;

    EMPTY_LIST   : constant List_Type := null;

-------------------------------------------------------------------
end List_Utilities;    --  END OF PACKAGE SPEC
-------------------------------------------------------------------
```

References

[1] *Reference Manual for the Ada Programming Language (MIL-STD-1815A)*
American National Standards Institute, 1983.

[2] *Military Standard Common APSE Interface Set (CAIS), proposed MIL-STD-CAIS*
KAPSE Interface Team and KAPSE Interface Team from Industry and Academia CAIS Working Group, 1985.

[3] Lindquist, Tim.
CAIS Operational Definition.
Technical Report, Arizona State University, 1985.

[4] *CAIS Rationale (DRAFT)*
1986.

[5] Gould, Inc.
(Untitled).
1986.

[6] Rebecca Bowerman.
Study of the Common APSE Interface Set (CAIS).
Technical Report 85W00537, The MITRE Corporation, 1820 Dolly Madison Boulevard, McLean, VA 22102, 1985.

[7] Oberndorf, Patricia A. (Chairman).
Kernel Ada Programming Support Environment (KAPSE) Interface Team: Public Report, Volume I.
Technical Report, Naval Ocean Systems Center, 1982.
Technical Document 509.

[8] KIT/KITIA.
DoD Requirements and Design Criteria for the Common APSE Interface Set (CAIS).
Technical Report, Ada Joint Program Office, 1985.
Prepared and Approved by the KAPSE Interface Team (KIT) and the KIT-Industry-Acedemia (KITIA) for the Ada Joint Program Office.

[9] *CAIS Reader's Guide (DRAFT)*
1985.

[10] Charles W. McKay and Rodney L. Bown.
Ada Run Time Support Environments and a Common APSE Interface Set.
In *AIAA/ACM/NASA/IEEE Computers in Aerospace V Conference*, pages 233-237. October, 1985.

[11] Buxton, John N.
Requirements for Ada Programming Support Environments (Stoneman)
Washington, D.C., 1980.

[12] *Trusted Computer System Evaluation Criteria, CSC-STD-001-83*
National Computer Security Center, 1983.

[13] B. Jackson and J. Salasin.
Preliminary Requirements for the Army WWMCCS Information System (AWIS).
Technical Report WP-84W00035, Mitre Corporation, 1984.
Working Paper.

Distribution List for IDA Paper P-1925

NAME AND ADDRESS	NUMBER OF COPIES
Sponsor	
LCDR Ronald Owens Ada Joint Program Office 1211 Fern Street, C-107 Arlington, VA 22202	5
Other	
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
IIT Research Institute 4550 Forbes Blvd., Suite 300 Lanham, MD 20706 (AJPO documents only)	1
Others	
Mr. Herman Fischer Litton Data Systems MARK V 16400 Ventura Blvd. Suite 303 Encino, CA 91436	1
Mr. Tim Harrison Software Productivity Consortium SPC Building 2214 Rock Hill Road Herndon, VA 22070	1
Mr. Tim Lindquist Computer Science Department Arizona State University Tempe, AZ 85287	1
Mr. Hans Mumm Naval Ocean Systems Command Code 423 San Diego, CA 92152	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Erhard Ploedereder Tartan Laboratories 477 Melwood Avenue Pittsburgh, PA 15213	1
Mr Carl Schmiedekamp Naval Air Development Center Code 5033 Warminster, PA 18974	1
Mr. Ray Syzmanski AFWAL/AAAF Wright Patterson AFB Dayton, OH 45433	1
Mr. Richard Thall SofTech, Inc. 460 Totten Pond Road Waltham, MA 02254-9197	1
Mr. Pete Carr Gould, Inc. C.S.D. 6901 West Sunrise Blvd. Fort Lauderdale, FL 33321	1
Mr. Mike Doub Manager, Ada Project Development Gould, Inc. 6901 West Sunrise Blvd. Fort Lauderdale, FL 33321	1
Mr. Robert Stevenson Manager, CAIS Project Gould Inc. C.S.D. 6901 West Sunrise Blvd. Fort Lauderdale, FL 33321	1
Mr. Jeff Vermette IBM (182/3J80) 18100 Frederick Pike Gaithersburg, MD 20879	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Harvey Burkett IBM (182/3J80) 18100 Frederick Pike Gaithersburg, MD 20879	1
Ms. Helen Gill MITRE Corporation 7525 Colshire Drive McLean, VA 22102	1
Mr. Chuck Howell MITRE Corporation 7525 Colshire Drive McLean, VA 22102	1
Ms. Robbie Hutchinson MITRE Corporation 7525 Colshire Drive McLean, VA 22102	1
Mr. Mike McClimens MITRE Corporation 7525 Colshire Drive McLean, VA 22102	1
Ms. Tana Reagan MITRE Corporation 7525 Colshire Drive McLean, VA 22102	1
Mr. Dan Burton Software Engineering Institute Carnegie-Mellon University Pittsburgh, PA 15213	1
Mr. Robert Ellison Software Engineering Institute Carnegie-Mellon University Pittsburgh, PA 15213	1
Mr. Peter Feiler Software Engineering Institute Carnegie-Mellon University Pittsburgh, PA 15213	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Dan Miller Software Engineering Institute Carnegie-Mellon University Pittsburgh, PA 15213	1
Mr. William G. Wood Software Engineering Institute Carnegie-Mellon University Pittsburgh, PA 15213	1
Mr. Mike Liu Department of Computer and Information Science Ohio State University 2036 Neil Avenue Mall Columbus, OH 43210	1
Mr. John Stankovic Dept. of Computer Science Carnegie-Mellon University Pittsburgh, PA 15213	1

CSED Review Panel

Dr. Dan Alpert, Director Program in Science, Technology & Society University of Illinois Room 201 912-1/2 West Illinois Street Urbana, Illinois 61801	1
Dr. Barry W. Boehm TRW Defense Systems Group MS R2-1094 One Space Park Redondo Beach, CA 90278	1
Dr. Ruth Davis The Pymatuning Group, Inc. 2000 N. 15th Street, Suite 707 Arlington, VA 22201	1

NAME AND ADDRESS	NUMBER OF COPIES
Dr. C.E. Hutchinson, Dean Thayer School of Engineering Dartmouth College Hanover, NH 03755	1
Mr. A.J. Jordano Manager, Systems & Software Engineering Headquarters Federal Systems Division 6600 Rockledge Dr. Bethesda, MD 20817	1
Mr. Robert K. Lehto Mainstay 302 Mill St. Occoquan, VA 22125	1
Dr. John M. Palms, Vice President Academic Affairs & Professor of Physics Emory University Atlanta, GA 30322	1
Mr. Oliver Selfridge 45 Percy Road Lexington, MA 02173	1
Mr. Keith Uncapher University of Southern California Olin Hall 330A University Park Los Angeles, CA 90089-1454	1
IDA	
General W.Y. Smith, HQ	1
Mr. Philip Major, HQ	1
Dr. Robert E. Roberts, HQ	1
Dr. John F. Kramer, CSED	1
Dr. Robert I. Winner, CSED	1
Ms. Anne Douville, CSED	1
Mr. Terry Mayfield, CSED	1
Mr. Clyde Roby, CSED	5
Ms. Sylvia Reynolds, CSED	2
IDA Control & Distribution Vault	3.